

In re Application of GOLDS et al.
Serial No. 09/768,098

REMARKS

The Office action has been carefully considered. The Office action explicitly rejected claim 24 as being directed to non-statutory subject matter and tacitly rejected claims 21-23 and 25 for the same reason. The Office action rejected claims 11, 13, 16-18, and 20 under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 6,148,336 to Thomas et al. ("Thomas"). Further, the Office action rejected claims 1-10, 19, and 21-25 under 35 U.S.C. § 103(a) as being unpatentable over Thomas in view of *Router Plugins, Asosoftware Architecture for Next Generation Routers* by Decasper et al. ("Decasper"). Finally, the Office action rejected claims 12, 14, and 15 under 35 U.S.C. § 103(a) as being unpatentable over Thomas. Applicants respectfully disagree.

By present amendment, claims 21, 24, and 25 have been amended for clarification and not in view of the prior art. Applicants submit that the claims as filed were patentable over the prior art of record, and that the amendments herein are for purposes of clarifying the claims and/or for expediting allowance of the claims and not for reasons related to patentability. Reconsideration is respectfully requested.

Prior to discussing reasons why applicants believe that the claims in this application are clearly allowable in view of the teachings of the cited and applied references, a brief description of the present invention is presented.

The present invention is directed to a system and method for ordering software modules in a persistent order for execution. To this end, the present invention provides a mechanism whereby unique numeric values may be statically assigned

In re Application of GOLDS et al.
Serial No. 09/768,098

to software modules at the time that each of the software modules (e.g., filter drivers) may be developed. Each module's assigned numeric value may determine its position relative to other modules in a stack or other ordered configuration. In this manner, the order for any given set of filter drivers may be fixed, eliminating bugs and other problems that result from alternative orderings, and also significantly simplifying testing.

In one implementation, this static value (sometimes referred to as an "altitude" because stacks are typically represented vertically) may comprise a precision floating-point number. As a result, when new software modules may be developed, each module may (in an existing execution order) be assigned a number that will enable that software module to be positioned between any two existing software modules, since between any two real numbers there exists an infinite number of other real numbers. By way of example, if altitudes such as 0.1 and 0.2 are assigned to filter drivers A and C, if some filter driver B is developed that needs to be ordered between A and C, there will be an unused altitude available between A and C that can be assigned to B, e.g., 0.15. If some other filter needed to attach between B and C, there will always be an unused altitude between B and C (e.g., 0.18) that is available.

When applied to filter drivers, the drivers may be generally classified according to their type, e.g., (antivirus, quota, encryption), as it is already known where such classes should approximately attach. For example, if altitudes are assigned values in the range from 0.0 to 1.0, where higher values attach closer to the base file system (e.g., NTFS), antivirus products may be assigned an altitude in

In re Application of GOLDS et al.
Serial No. 09/768,098

the 0.2 to 0.3 range, quota drivers between 0.4 and 0.6, and encryption filters between 0.7 and 0.8. Moreover, drivers of the same type may also be ordered among one another within their general range, which may guarantee only one possible ordering in both testing and actual operation. Note that the above description is for example and informational purposes only, and should not be used to interpret the claims, which are discussed below.

§101 Rejections

The Office action rejected claims 21-25 as being directed to non-statutory subject matter. More specifically, the Office action contends that claim 24 is directed to a method that may be carried out in one's mind and also contends that claims 21-23 and 25 are directed to a computer-readable medium and goes further to suggest that the specification recites a limitation on the term computer-readable medium as a modulated signal or carrier wave that makes claims 21-23 and 25 non-statutory. Applicant respectfully disagrees.

Section 2106(IV)(B)(1)(a) of the MPEP states that functional descriptive material that is recorded on some computer-readable medium is structurally and functionally interrelated to the medium and is statutory since use of technology permits the function of the descriptive material to be realized. See *In re Lowry*, 32 F.3d 1579, 1583-84, 32 USPQ2d 1031, 1035 (Fed. Cir. 1994) (claim to data structure stored on a computer readable medium that increases computer efficiency held statutory) and *In re Warmerdam*, 33 F.3d at 1360-61, 31 USPQ2d at 1759 (claim to computer having a specific data structure stored in memory held

In re Application of GOLDS et al.
Serial No. 09/768,098

statutory product-by-process claim). Carrier waves and modulated signals are examples of data that may be interpreted by a computer (*i.e.*, a computer-readable medium) and may also be considered a product-by-process which is also statutory *per se* if the underlying process is statutory. Furthermore, the MPEP specifically states (section 2106(IV)(B)(1)(c)) that a signal claim directed to a practical application is statutory regardless of its transitory nature. See *O'Reilly*, 56 U.S. at 114-19; *In re Breslow*, 616 F.2d 516, 519-21, 205 USPQ 221, 225-26 (CCPA 1980). Recent court decisions have also held that "signals" are proper statutory subject matter. See *Arrhythmia Research Technology, Inc. v. Corazonix Corp.*, 958 F.2d 1053, 22 USPQ.2d 1033 (CCPA 1992)(wherein the court held as incorrect the view that "signals" are improper statutory subject matter simply because there may be nothing necessarily physical about "signals" and held that computer-program related inventions can be claimed in terms of "signals" because computers operate according to signals. In fact, anything that is being manipulated or transformed can typically be drafted in terms of "signals").

Notwithstanding this, claims 21, 24, and 25 have been amended to recite a computer-implemented method or a tangible computer-readable medium including, for example, when any signal may be loaded into a memory. Certainly a tangible computer-readable medium, *e.g.*, a memory, a computer disk, *etc.* is statutory subject matter. For at least these reasons, applicant requests that the §101 rejection of claims 21-25 be withdrawn.

In re Application of GOLDS et al.
Serial No. 09/768,098

§102 Rejections

Turning to the claims rejected under §102, claim 11 recites in a computer system, a mechanism comprising a plurality of software modules, each software module having a static assigned value indicative of a relative order, and an ordering mechanism configured to evaluate each static assigned value and to arrange the software modules for execution in a relative order determined by the assigned values, the order being deterministic and static.

The Office action rejected claim 11 as being anticipated by Thomas. More specifically, the Office action contends that Thomas teaches a plurality of software modules, each module having a static assigned value indicative of a relative order and an ordering mechanism configured to evaluate each value and to arrange the software modules for execution in a relative order determined by the static assigned values, the order being deterministic and static. Column 5, lines 7-53, column 6, lines 59-67, column 7, lines 45-53, column 8, lines 19-43 and column 10, line 42 to column 11, line 58 of Thomas is referenced. Applicants respectfully disagree.

Thomas is directed, generally, toward a system and method for filtering, sorting, and executing plugin network-service providers. More specifically, Thomas teaches an extensible service provider with an upper interface to a higher-level network-socket library and a lower interface to a network-transport layer that formats data for transmission over a network. A plugin manager controls the plugin network-service providers (classified as examining, blocking, modifying or

In re Application of GOLDS et al.
Serial No. 09/768,098

encrypting) such that the plugin network-service providers are sorted into the execution order that executes plugin network-service providers in the order: examining, blocking, modifying, and then encrypting functions, for outgoing data.

Thus the data is examined before modified, modified before encrypted, *etc.* In contrast, claim 11 recites each software module having a static assigned value indicative of a relative order. The static value, which may be assigned at the coding of the software module and is typically numeric, cannot change without recoding the software module (which may not simply be done in practice since a new software module may also be coded to replace the old). Thus, the assigned values for indicating a relative order of execution may be static and persistent. Thomas does not teach associating any value with any software module at any time. Thomas teaches classifying plug-ins modules according to a relative function, *i.e.*, examining, blocking, modifying or encrypting. Classifying plug-ins according to function is not the same as assigning a static value to each software module.

Even if one were to somehow construe classification of function as similar to assigning a static value, Thomas still embodies the same problems of any other conventional systems, as all plug-ins classified as examining plug-ins will be executed in a random and non-deterministic manner because the plug-ins are not assigned a static value that may be deterministically compared to other plug-ins with static values. Thus, the same problems will occur in that all examining plug-ins will be executed first among all plugins but randomly and undeterministically among just the plugins classified as examining plugins.

In re Application of GOLDS et al.
Serial No. 09/768,098

The static binding list of traffic filters attached to each plugin network-service provider, as further taught by Thomas, does not cure this deficiency. Traffic filters are merely used in Thomas to direct traffic to and from different resources in the system. By analyzing the protocol level of traffic, decisions may be made as to where network traffic may be routed. The protocol level of network traffic, however, is wholly unrelated to the function of a plugin module. Simply put, Thomas does not teach each software module having a static assigned value indicative of a relative order.

Furthermore, claim 11 recites an ordering mechanism configured to evaluate each static assigned value and to arrange the software modules for execution in a relative order determined by the assigned values, the order being deterministic and static. Once the software modules have been assembled, there may exist an order in which the software modules may be executed. This order may be deterministic (*i.e.*, known beforehand) and static (*i.e.*, will not change unless additional or different software modules are added). The system taught by Thomas, however, is completely silent as to ordering any execution of software modules within a classified function. Although, Thomas teaches executing examining plugins before blocking plugins, *etc.*, it is completely silent as to the order of execution wholly within the classified functions. Thus, among blocking plugins, execution is undeterministic and random. Furthermore, any future execution of the same sequence of plugins is not guaranteed to be the same, as there is no static value assigned to the plugins that determine a relative execution order. For at least

In re Application of GOLDS et al.
Serial No. 09/768,098

these reasons, applicants submit that claim 11 is patentable over the prior art of record.

Applicants respectfully submit that dependent claims 13, 16-18, and 20, by similar analysis, are allowable. Each of these claims depends either directly or indirectly from claim 11 and consequently includes the recitations of independent claim 11. As discussed above, Thomas fails to disclose or suggest the recitations of claim 11 and therefore these claims are also allowable over the prior art of record. In addition to the recitations of claim 11 noted above, each of these dependent claims includes additional patentable elements.

For example, claim 13 recites the mechanism of claim 11 wherein the software modules comprise filter drivers. The Office action contends that Thomas teaches filter drivers. This is erroneous logic as the Office action has originally equated a software module with a plugin which is different from the filter mechanism as taught by Thomas. The Office action cannot then construe the filters of Thomas to also be plugin modules. A filter driver may be a software module that contains executable language for allowing a computer system to work with a particular filter. A filter driver and a filter are not the same. For at least this additional reason, applicants submit that claim 13 is allowable over the prior art of record.

§103 Rejections

Turning to the first independent claim rejected under §103, claim 1 recites in a computer system, a method, comprising maintaining static assigned values in

In re Application of GOLDS et al.
Serial No. 09/768,098

association with software modules, each software module having a static assigned value, the assigned values having a relative order and there being an unassigned value between any two assigned values, and executing the software modules in an order determined by each of the assigned values, the order being deterministic and static.

The Office action rejected claim 1 as being unpatentable over Thomas in view of Decasper. More specifically, the Office action contends that Thomas teaches maintaining assigned values in association with software modules, each software module having an assigned value. Further, the Office action contends that Thomas teaches the assigned values having a relative order, and executing the software modules in an order determined by each of the assigned values. Column 5, lines 7-53, column 6, lines 59-67, column 7, lines 45-53, column 8, lines 19-43 and column 10, line 42 to column 11, line 58 of Thomas is referenced. The Office action concedes that Thomas fails to teach having unassigned values between any two assigned values of software modules. However, the Office action contends that Decasper does disclose such subject matter by teaching dynamic loading and unloading of software modules that have static assigned values by reference to TCP communications. Page 235, table 1 of Decasper is referenced.

The Office action concludes that it would be obvious to one skilled in the art at the time of the invention to combine the teachings of Thomas with the teachings of Decasper because the ability to bind different plug-ins to individual flows in the same run-time environment is desirable. Applicants respectfully disagree.

In re Application of GOLDS et al.
Serial No. 09/768,098

To establish *prima facie* obviousness of a claimed invention, all of the claim recitations must be taught or suggested by the prior art; (*In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974)), and "all words in a claim must be considered in judging the patentability of that claim against the prior art;" (*In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970)). Further, if prior art, in any material respect teaches away from the claimed invention, the art cannot be used to support an obviousness rejection. *In re Geisler*, 116 F.3d 1465, 1471, 43 USPQ2d 1362, 1366 (Fed Cir. 1997). Moreover, if a modification would render a reference unsatisfactory for its intended purpose, the suggested modification / combination is impermissible. See MPEP § 2143.01

Applicants submit that the Office action has failed to establish a *prima facie* case for obviousness. As discussed above, Thomas is directed, generally, toward a system and method for filtering, sorting, and executing plugin network-service providers. Plugin network-service providers are classified by function into classes of examining, blocking, modifying, and then encrypting functions. Classifying a plugin by class is not the same as assigning a static value to a software module. Being able to assign a static value to a software module, as recited in claim 1, presumes that any number may be assigned, reassigned, changed and removed. The system of Thomas may not classify the function of a plugin in any class other than the class that it is. For example, in the present invention, a software module, such as a blocking plugin, may be assigned a value of 0.15 which may indicate execution earlier or may be assigned a static value of 0.97 which may indicate execution later, etc. However, in Thomas, the blocking plugin may only be

In re Application of GOLDS et al.
Serial No. 09/768,098

classified one way, *i.e.*, as a blocking plugin, and may only be executed in an order when other blocking plugins are executed. As such, Thomas simply does not teach maintaining static assigned values in association with software modules, each software module having a static assigned value as recited in claim 1.

The teachings of Decasper do not cure this deficiency as Decasper is wholly concerned with redirecting network traffic based on a set of traffic filters wherein protocol dictates the resource in which network traffic will be directed. Directing traffic based on TCP/IP parameters is not the same as assigning static value to software modules wherein the static assigned values are indicative of a relative execution order. As such, Thomas and Decasper, whether considered individually or in any combination with each other or any other prior art of record, do not teach or suggest maintaining static assigned values in association with software modules, each software module having a static assigned value as recited in claim 1.

Furthermore, claim 1 recites the assigned values having a relative order and there being an unassigned value between any two assigned values. Thomas does not teach assigning values to software modules as discussed above and the Office action explicitly acknowledges that Thomas does not teach an unassigned value between any two assigned values. Decasper does not cure these deficiencies as Decasper is wholly concerned with re-routing traffic in a networked environment using traffic filters and typical TCP/IP networking tenets. For one, typical IP addressing does not, in fact, always have an unassigned value between any two assigned values. As an example, the IP addresses 192.92.233.10 and

In re Application of GOLDS et al.
Serial No. 09/768,098

192.92.233.11 do not have any IP address between them. These IP addresses are statically and deterministically consecutive and there may not be another IP address between them. Thus, the very nature of IP addressing does not allow for any new static assigned values to exist between any two previous static assigned values. Neither Thomas, nor Decasper teaches an unassigned value between any two assigned values among static assigned values for software modules as recited in claim 1.

For at least these reasons, applicants submit that claim 1 is patentable over the prior art of record.

Claims 2-10 were rejected as unpatentable over Thomas in view of Decasper. Applicants respectfully submit that dependent claims 2-10, by similar analysis, are allowable. Each of these claims depends either directly or indirectly from claim 1 and consequently includes the recitations of independent claim 1. As discussed above, Thomas and Decasper, whether considered alone or in any permissible combination with each other or any other prior art of record, fail to teach or suggest the recitations of claim 1 and therefore dependent claims 2-10 are also allowable over the prior art of record. In addition to the recitations of claim 1 noted above, each of these dependent claims includes additional patentable elements.

For example, claim 3 recites the software modules comprise filter drivers, and wherein calling the software modules includes passing file system requests thereto. The Office action contends that a filter driver reads on filter as taught by Thomas. This is blatantly incorrect. A filter driver may be a software module that

In re Application of GOLDS et al.
Serial No. 09/768,098

contains executable language for allowing a computer system to work with a particular filter. For at least this additional reason, applicants submit that claim 3 is allowable over the prior art of record.

Claims 12, 14, and 15 were rejected as unpatentable over Thomas in view of Decasper and claim 19 was rejected as being unpatentable over Thomas alone. Each of these claims depends either directly or indirectly from claim 11 and consequently includes the recitations of independent claim 11. Applicants respectfully submit that dependent claims 12, 14, 15, and 19, by similar analysis, are allowable. As discussed above, Thomas fails to teach or suggest the recitations of claim 11. Neither Thomas, Decasper, nor any other prior art of record, whether considered alone or in any permissible combination, teach or suggest the recitations of claim 11 and therefore dependent claims 12, 14, 15, and 19 are also allowable. In addition to the recitations of claim 11 noted above, each of these dependent claims includes additional patentable elements.

Turning to the next independent claim, amended claim 21 recites a tangible computer-readable medium having computer-executable instructions, comprising maintaining static assigned values in association with filter drivers, each filter driver having an assigned value, the assigned values having a relative order and there being an unassigned value between any two assigned values, and executing the filter drivers in an order determined by each of the assigned values, the order being deterministic and static.

The Office action rejected claim 21 as being unpatentable over Thomas in view of Decasper. The Office action cites the exact same reasoning and citations

In re Application of GOLDS et al.
Serial No. 09/768,098

to Thomas and Decasper as detailed with respect to the rejection of claim 1.

Applicants respectfully disagree.

Applicants submit that the Office action has failed to establish a *prima facie* case for obviousness. Unlike Thomas, claim 21 recites maintaining static assigned values in association with filter drivers. The static value, which may be assigned at the coding of the software module, cannot change without recoding the software module. Thus, the assigned values for indicating a relative order of execution are static and persistent. This is different than Thomas in that Thomas does not teach assigning values but rather classifying by function. Consequently, Thomas simply does not teach or suggest maintaining static assigned values in association with filter drivers as recited in claim 21.

Furthermore, claim 21 recites executing the filter drivers in an order determined by each of the assigned values, the order being deterministic and static. That is, once the filter drivers have been assembled, there exists one and only one order in which the filter drivers may be executed and this order is deterministic (*i.e.*, known before-hand) and static (*i.e.*, will not change unless additional or different filter drivers are added). The system taught by Thomas, however, is completely silent as to ordering any execution of plugin within a similar classification. That is, among blocking plugins, there does not exist a singular deterministic execution order, in contrast to the plain claim language.

Further yet, the Office action contends that a filter driver reads on filter as taught by Thomas. This is flatly wrong. A filter driver may be a software module

In re Application of GOLDS et al.
Serial No. 09/768,098

that contains executable language for allowing a computer system to work with a particular filter. A filter driver is not a filter as taught by Thomas.

Moreover, the Office action concludes that the recitations of claim 21 would be obvious to one skilled in the art at the time of the invention because the ability to bind different plug-ins to individual flows in the same run-time environment is desirable. This is overly broad and conclusory, among other reasons because Decasper teaches a filtering system based on IP addressing that is wholly unconcerned with the nature of the network traffic, *i.e.*, whether an examining plugin or a blocking plugin is being passed. Such broad, conclusory statements do not come close to adequately addressing the issue of motivation to combine, are not evidence of obviousness, and therefore are improper as a matter of law. *In re Dembiczak*, 175 F.3d 994, 999, 50 USPQ2d 1614, 1617 (Fed. Cir. 1999). In short, Thomas and Decasper, whether considered individually or in any combination with each other or any other prior art of record, do not teach or suggest an unassigned value between any two assigned values as recited in claim 21.

For at least these reasons, applicants submit that claim 21 is patentable over the prior art of record.

Applicants respectfully submit that dependent claims 22 and 23, by similar analysis, are allowable. Each of these claims depends directly from claim 21 and consequently includes the recitations of independent claim 21. As discussed above, Thomas and Decasper, whether considered individually or in any permissible combination with each other or any other prior art of record, fail to teach or suggest the recitations of claim 21 and therefore claims 22 and 23 are

In re Application of GOLDS et al.
Serial No. 09/768,098

also allowable. In addition to the recitations of claim 21 noted above, each of these dependent claims includes additional patentable elements.

Turning to the last independent claim, amended claim 24 recites a computer-implemented method, comprising classifying software modules into groups based on types thereof, assigning each software module a static value based on its group, each assigned value having a relative order that is deterministic and static and there being an unassigned value between any two assigned values, and maintaining an association between each software module and its assigned value.

The Office action rejected claim 24 as being unpatentable over Thomas in view of Decasper. More specifically, the Office action contends that Thomas teaches classifying software modules into groups based on types thereof, assigning each software module a static value based on its group, each assigned value having a relative order that is deterministic and static, and maintaining an association between each software module and its assigned value. Column 5, lines 7-53, column 6, lines 59-67, column 7, lines 45-53, column 8, lines 19-43 and column 10, line 42 to column 11, line 58 of Thomas is referenced.

The Office action acknowledges that Thomas fails to teach having unassigned values between any two assigned values of software modules. However, the Office action contends that Decasper teaches dynamic loading and unloading of software modules that have static assigned values by reference to TCP communications. Page 235, table 1 of Decasper is referenced. The Office action concludes that it would be obvious to one skilled in the art at the time of the

In re Application of GOLDS et al.
Serial No. 09/768,098

invention to combine the teachings of Thomas with the teachings of Decasper because the ability to bind different plug-ins to individual flows in the same run-time environment is desirable. Applicants respectfully disagree.

Applicants submit that the Office action has failed to establish a *prima facie* case for obviousness. As discussed above, Thomas is directed, generally, toward a system and method for filtering, sorting, and executing plugin network-service providers. As such, plugin network-service providers are classified by function into classes of examining, blocking, modifying, and then encrypting functions.

Classifying a plugin by class is not the same as assigning a static value to a module of a group of software modules.

The system of Thomas may not classify the function of a plugin in any class other than the class that it is. For example, in the present invention a software module, such as a blocking plugin, may be assigned a value of 0.15 which may indicate execution earlier among all grouped as a blocking plugin or may be assigned a static value of 0.97 which may indicate execution later along with other blocking plugins, etc. However, in Thomas, the blocking plugin may only be classified one way, *i.e.*, as a blocking plugin, and may only be executed in an order when other blocking plugins are executed, *i.e.*, after examining plugins and before modifying plugins. As such, Thomas simply does not teach classifying software modules into groups based on types thereof, assigning each software module a static value based on its group as recited in claim 24.

The teachings of Decasper do not cure this deficiency as Decasper is wholly concerned with redirecting network traffic based on a set of traffic filters wherein

In re Application of GOLDS et al.
Serial No. 09/768,098

protocol dictates the resource in which network traffic will be directed. Directing traffic based on TCP/IP parameters is not the same as assigning static value to software modules wherein the static assigned values are indicative of a relative execution order. As such, Thomas and Decasper, whether considered individually or in any combination with each other or any other prior art of record, do not teach or suggest classifying software modules into groups based on types thereof, assigning each software module a static value based on its group as recited in claim 24.

Furthermore, claim 24 recites the assigned values having a relative order and there being an unassigned value between any two assigned values. Thomas does not teach assigning values to software modules as discussed above and the Office action explicitly acknowledges that Thomas does not teach an unassigned value between any two assigned values. Decasper does not cure these deficiencies as Decasper is wholly concerned with re-routing traffic in a networked environment using traffic filters and typical TCP/IP networking tenets. As discussed above, the IP addresses 192.92.233.10 and 192.92.233.11 do not have any IP address between them. Thus, the very nature of IP addressing does not allow for any new static assigned values to exist between any two previous static assigned values. Neither Thomas, nor Decasper teaches an unassigned value between any two assigned values among static assigned values for software modules as recited in claim 24.

For at least these reasons, applicants submit that claim 24 is allowable over the prior art of record.

In re Application of GOLDS et al.
Serial No. 09/768,098

Applicants respectfully submit that dependent claim 25 by similar analysis, is allowable. This claim depends directly from claim 24 and consequently includes the recitations of independent claim 24. As discussed above, Thomas and Decasper, whether considered individually or in any permissible combination with each other or any other prior art of record, fail to teach or suggest the recitations of claim 24 and therefore claim 25 is also allowable over the prior art of record. For at least these additional reasons, applicants submit that all the claims are patentable over the prior art of record. Reconsideration and withdrawal of the rejections in the Office action is respectfully requested and early allowance of this application is earnestly solicited.

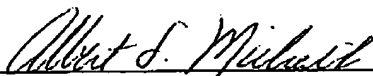
In re Application of GOLDS et al.
Serial No. 09/768,098

CONCLUSION

In view of the foregoing remarks, it is respectfully submitted that claims 1-25 are patentable over the prior art of record, and that the application is in good and proper form for allowance. A favorable action on the part of the Examiner is earnestly solicited.

If in the opinion of the Examiner a telephone conference would expedite the prosecution of the subject application, the Examiner is invited to call the undersigned attorney at (425) 836-3030.

Respectfully submitted,



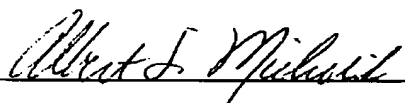
Albert S. Michalik, Reg. No. 37,395
Attorney for Applicants
Law Offices of Albert S. Michalik, PLLC
704 - 228th Avenue NE, Suite 193
Sammamish, WA 98074
(425) 836-3030
(425) 836-8957 (facsimile)

In re Application of GOLDS et al.
Serial No. 09/768,098

CERTIFICATE OF FACSIMILE TRANSMISSION

I hereby certify that this Amendment, along with transmittal, petition for extension of time, credit card payment form, and facsimile cover sheet, are being transmitted by facsimile to the United States Patent and Trademark Office in accordance with 37 C.F.R. 1.6(d) on the date shown below:

Date: November 11, 2005



Albert S. Michalik

2630 second amendment